UCRL-WEB-200048

EZOUTPUT

Table of Contents

Preface

Scope: EZOUTPUT explains how to send files from LC computers to network printers (using LP, LPR, and APR), how to monitor and delete print requests, and how to control their output characteristics (using special format options or extra file preprocessors). It also explains how to convert graphics formats into PostScript for printing. Basic LC-supported file-transfer tools often needed to prepare or manage output, including FTP, NFT, SCP, SFTP and HTAR, are also explained. The dedicated (FIS) service for passing files between the secure and open networks (either direction), as well as key features of HOPPER, LC's visual controller for other file-transfer programs, are also explained.

Availability: The programs described here run under any LC system unless otherwise noted.

Consultant: If you require support, contact the LC customer service and support hotline at 925-422-4531 (open e-mail: lc-hotline@llnl.gov, SCF e-mail: lc-hotline@pop.llnl.gov).

Printing: The print file for this document can be found at:

```
OCF: https://computing.llnl.gov/LCdocs/ezoutput/ezoutput.pdf
SCF: http://www.llnl.gov/LCdocs/ezoutput/ezoutput_scf.pdf
```

Introduction

This manual provides a basic guide for generating output and transferring files on LC computers. Its goal is to include the most important and relevant tools, techniques, and features, while making them easy to find and learn by avoiding more esoteric features or comprehensive descriptions of every possible feature.

Section 1: Outlines naming and locating tips for local network printers and introduces local tools (such as LISTPQ) for discovering and taking advantage of LC's standardized print-queue names.

Section 2: Explains how to convert older graphics file formats into PostScript for printing with LP, LPR, or APR.

Section 3: Explains how to print files using LP, LPR, or APR, whichever is most appropriate for your situation.

Section 4: Compares the utilities SCP, FTP, SFTP and NFT, and explains the basic features of each for between-machine file transfers, as well as the key features of LC's visual file-transfer controller called HOPPER.

Section 5: Introduces LC's special open/secure File Interchange Service (FIS).

Reference manuals are available to provide detailed technical instructions for the tools and techniques introduced in EZOUTPUT, including manuals that support FTP (with SFTP) (URL: https://computing.llnl.gov/LCdocs/ftp), NFT (URL: https://computing.llnl.gov/LCdocs/nft), HTAR (URL: https://computing.llnl.gov/LCdocs/htar), and FIS (URL: https://computing.llnl.gov/LCdocs/fis). You may also want to consult the basic guide for using local directories and for general file-handling software at LC, called EZFILES (URL: https://computing.llnl.gov/LCdocs/ezfiles).

If your concern is efficiently managing I/O for science application codes as they run in LC's parallel computing environments, consult the separate I/O Guide for LC (URL: https://computing.llnl.gov/LCdocs/ioguide). For LC-appropriate and very scalable *graphics* output, such as exploratory visualizations of local simulation data, consult the draft TeraScale Browser Manual (URL: https://computing.llnl.gov/LCdocs/tsb).

Section 1: Using Local Printers

Print Queue Names

BASIC NAME.
Each printer is associated with one basic print queue whose name always has the form p*nnn*, where *nnn* is a three digit number that uniquely identifies that printer. Examples are p005 and p123.

FEATURE NAMES.
Most printers also have additional print queue names that derive from the basic name and that invoke special processing features. The most common feature names include:

- *duplex* (two-sided) printing, with the suffix '_dup' appended to the basic name. Examples are p005_dup and p1233_dup.
- *transparency* (viewgraph) printing, with the suffix '_trans' appended to the basic name. Examples are p005_trans and p123_trans.

LOCATION NAMES.
Each printer also has one print queue name that reveals its physical location by prefixing the basic name with a

- *building* identifier (uppercase B, e.g. B453)
- *room* identifier (uppercase R, e.g. R1000)

Examples of location names are B111_R367_p122 and B113_R2024_p020. Color printers have the extra prefix C_ inserted between the room number and the basic name in their location name (e.g., B111_R_367_C_p122).

Thus a single physical LC printer typically has four associated print-queue names, which follow the pattern exemplified here:

```
p020 [basic name]
p020_dup
p020_trans
B113_R2024_C_p020
```

Discovering Printers

LISTPQ (Location-Based Queue Discovery)

On LC machines, LISTPQ helps you discover the unique basic name of a printer or the print queue(s) associated with the physical printer in a specified location by typing

`listpq` [*`options`*]

where *options* let you specify a building, room, floor, or model for LISTPQ's report.

WARNINGS:

(1) By default, with no options, LISTPQ lists *all* LC print queue names but without descriptive comments, a result that is seldom useful. A better strategy is to specify a location and then include "-i report" to get some descriptive details.

(2) The *order* of your options can affect how quickly LISTPQ generates your report. Specify your most restrictive options (such as -b) first, then let less restrictive ones (such as -ink) follow for faster results.

OPTIONS:

-b[uilding] *num*

lists all print queues that feed printers in the LLNL building numbered *num* (digits only, such as -b 121).

-r[oom] *string* lists all print queues that feed printers in the LLNL room specified by *string* (either the digits of a room number (144) or a room's name (lobby)).

-i[nfo] report generates a multi-column table each row of which describes one print queue that meets the criteria set by the other, location-specifying options. The description includes building, room, floor, ink, and printer model. Replace the literal REPORT with DELIMITED to get a colon-delimited list of rows (no spacing) instead; invoke the -NOHEADER option to omit the column heads in the REPORT table.

-ink *colors* lists all print queues that feed printers that offer *colors* ink, where *colors* is either BW or COLOR (uppercase or lowercase).

-mo[del] *string* lists all print queues that feed printers of the specified model, which can be any identifying string (such as LASERJET, any case).

-[no]header toggles the column headings in the -i report output (-header is the default, and much less confusing, choice).

-old adds a column to the -i report output that reveals the former (nonstandardized) name of each listed print queue.

EXAMPLE:

To list a row of descriptive features (with explanatory headings) for each (and only) color print queue feeding printers located in LLNL's Building 116, use the execute line

```
listpq -b 116 -ink color -i report
```

Omitting -ink generates a (longer, more inclusive) report for all B-116 print queues more quickly.

LPSTAT

On LC machines, LPSTAT gives you the current status of a physical printer but can also be used to help you discover the unique basic name of a printer or the print queue(s) associated with the physical printer in a specified location by typing

```
lpstat [options]
```

where *options* let you specify other parameters to include in LPSTAT's report.

NOTE:

(1) By default, with no options, LPSTAT lists print jobs queued by the current user.

OPTIONS:

-E	forces encryption when connecting to the server.
-R	shows the ranking of print jobs.
-U username	specifies an alternate username.
-W which-jobs	specifies which jobs to show, *completed* or *not-completed* (the default). This option must appear before the -o option and/or any printer names, otherwise the default value will be used in the request to the scheduler.
-a [printers(s)]	shows the accepting state of printer queues. If no printers are specified, then all printers are listed.
-c [class(es)]	shows the printer classes and the printers that belong to them. If no classes are specified, then all classes are listed.
-d	shows the current default destination.
-h server [:port]	specifies an alternate server.
-l	shows a long listing of printers, classes, or jobs.
-o [destinations(s)]	shows the jobs queue on the specified destinations. If no destinations are specified, all jobs are shown.
-p [printer(s)]	shows the printers and whether or not they are enabled for printing. If no printers are specified, then all printers are listed.

-r shows whether the CUPS server is running

-s shows a status summary, including the default destination, a list of classes and their member printers, and a list of printers and their associated devices. This is equivalent to using the *-d*, *-c*, and *-y* options.

-t shows all status information. This is equivalent to using the *-r*, *-d*, *-c*, *-v*, *-a*, *-p*, and *-o* options.

-u [users(s)] shows a list of print jobs queued by the specific users. If not users are specified, then only print jobs queued by the current user are listed.

EXAMPLE:

```
lpstat -a | grep -i B111_R4
```

This would result in the following output:

```
zeus286@arnold:lpstat -a | grep -i B111_R4
B111_R467_C_p011 accepting requests since Fri May 23 13:27:32 2008
B111_R467_p119 accepting requests since Fri May 23 13:27:32 2008
```

From this, we know that there are two printers in B111, R467. The "C" in the building/room name means it is a color printer. By using the lpstat -a and grep you can then identify the "p" name.

Using **`lpstat -a`** again, we can see if there are other queues associated with the "p" name.

EXAMPLE:

```
lpstat -a | grep p119
```

This would result in the following output:

```
zeus286@arnold:lpstat -a | grep p119
B111_R467_p119 accepting requests since Fri May 23 13:27:32 2008
p119 accepting requests since Fri May 23 12:49:54 2008
p119_dup accepting requests since Fri May 23 12:49:54 2008
```

You can see that p119_dup has been configured for duplex printing by default, but there may be some exceptions where the printer queue names don't follow the above convention.

Environment Variables for Default Printers

On machines with LPR or APR, you can set the UNIX environment variable PRINTER (always uppercase) to specify the name of a default printer. Then, if you run LRP or APR without the -P option, output will go to the default printer you specified.

Likewise, on machines with LP, you can set the UNIX environment variable LPDEST (uppercase) to specify the name of a default printer. Then, if you run LP without the -d option, output will go to the default printer you specified.

To set either of these environment variables for your current login session while you run the C shell on any UNIX machine, type

```
setenv PRINTER printername
```

or

```
setenv LPDEST printername
```

to automatically set either either environment variable whenever you log in, include the appropriate one of these lines in the *.cshrc run-control* file that resides in your home directory.

To report the current value of an environment variable (such as PRINTER) while in the C shell, use PRINTENV:

```
printenv PRINTER
```

For a comparative explanation of using environment variables to manage program behavior in general on LC machines, see LC's Environment Variables (URL: https://computing.llnl.gov/LCdocs/ev) user guide.

Section 2: Printing Files

Graphics files and other file formats must be converted to a text or PostScript file using one of the conversion tools discussed in "Converting File Formats for Printing" (page 19).

Output Software Compared

The two standard UNIX tools for printing files are LP and LPR. Both have their own distinct related routines for monitoring print queues and for deleting pending print jobs. However, neither LP nor LPR meets local needs for automatically labeling each page of printed output in a secure environment. Hence, a locally enhanced tool called APR adds this feature (among others) on secure systems at LC.

The table below summarizes the user-relevant differences between LP, LPR, and APR. (See the "Special Cases" (page 15) section for details on using APR in some unusual ways and for invoking extra options to control the details of output format.)

	LP	**LPR**	**APR**
Available on:	Linux: all clusters Open clusters IBM: UP, UBGL	Linux: all clusters Open clusters IBM: UP, UBGL	Linux: SCF clusters SCF clusters IBM: Purple
Related program to:			
Show print queue	lpstat	lpq	lpq
Delete print job	cancel	lprm	lprm
Environment variable checked for default printer	LPDEST	PRINTER	PRINTER
Supports security page labels?	no	no	yes
Accepts PR preprocessing?	yes	yes	no
Typical command lines:	lp -d *prname* *flname* lpstat	lpr -P *prname* *flname* lpq -P *prname*	apr -P *prname* -Su *flname* lpq -P *prname*
With options to:			
Specify printer	-d *prname*	-P *prname*	-P *prname*
Label banner page	-t *string*	-J *string*	-- -J *string*
Make *num* copies	-n *num*	-# *num*	-# *num*
Add security labels	none	none	-S *level*
Control output format	-o *feature*	-o *feature*	none

Security Levels for APR

APR is the only output routine that labels the top and bottom of every printed page with a security-level banner of the form

```
--- XXX --- Mon May 11 10:30:45 PDT 2008 --- XXX ---
```

which includes the date printed and a standard security label XXX. You specify the security level of the file(s) you print by using APR's -S *level* option, which has no default value so you must explicitly state the level, and APR assigns a corresponding security label for XXX according to this chart:

Security Level Specified	**Label Output**
s, S, srd, SRD, 5	Secret Restricted Data
c, C, crd, CRD, 4	Confidential Restricted Data
u, U, uncl, UNCL, 1	Unclassified

If you direct APR's output to a file (using the -p option), then each PostScript page in that file also bears the security-level banner that you specified with -S *level*.

NOTE: Classified print queues for output devices in LC's classified computer center (B453) are disabled on nights and weekends as a security precaution. See the Night and Weekend Output (page 18) section for more information.

Examples

Submitting, Monitoring, Deleting a Print Request (Open)

GOAL: On an open machine, to send two text files (test1 and test2) to a printer, check the print queue, then (supposing you changed your mind) delete the print request while it waits.

STRATEGY: On Atlas, use LPR to send the files to a printer (e.g. p280), use LPQ to check the queued files, and use LPRM to remove them from the print queue. You can remove queued files by using the JOB name assigned to them, NOT the original file name(s), but since the printer processes files very quickly, especially when there are no other jobs in the queue, this might be difficult. Also, LPR does not return the job name when you submit a print request, so you must run LPQ to discover the job name (e.g. 32) if you need it. If you had run LPR twice to submit each file separately, you could later run LPRM to delete either file separately.

```
lpr -P p280 test1 test2
[no visible response from LPR]
lpq -P p280

 p280 is ready and printing
 Rank   Owner      Job   Files                 Total Size
 active trg        32    test1, test2          7542 bytes

lprm -P p280 32
 lpr: ps280-32 dequeued
```

Printing a Classified File with Appropriate Labels (Secure)

GOAL: To send Confidential Restricted Data (CRD) file test3 and Secret Restricted Data (SRD) file test4 to appropriate printers with appropriate page labels.

STRATEGY: Use APR to label the test3 pages as CRD (with -S c) and send it to the printer called b113r2040crd. Then use APR again to label the test4 pages as SRD (with -S s) and send it to the printer called b113r2222secret.
The long names of the printers reveal both their location and their highest-allowed security level. LPQ and LPRM support APR in the same way that they support LPR in the previous example.

```
apr -P b113r2040crd -S c test3
apr -P b113r2222secret -S s test4
```

Special Cases

Printing to a File

On a secure machine offering APR, you have the option to "print to a file." This lets you save a document as a PostScript file with your specified security banner at the top and bottom of each page. Use the lowercase -p option instead of the usual uppercase -P to name the output file:

```
apr -p myout.ps -S s project37
```

You can then preview this output file with PostScript tools (such as GHOSTVIEW) before printing it with LPR (since it is already security-labeled) or storing it. Neither LP nor LPR offers a corresponding option to "print to a file."

Printing Big Files

By default, LP links the file you want to print to an entry in the printer queue instead of actually copying the file, which saves disk space--a lot if the file is very large. You can force LP to copy the file to the printer queue instead of linking if you use the -c option on LP's command line.

Unfortunately, LPR reverses this behavior. By default, LPR copies your private files to a spooling directory where they wait to print. This duplication can use a lot of disk space or even prevent printing a very large file if insufficient space remains for the extra copy. You can force LPR to use links instead of actual copies to save space when printing big files if you use the -s option on LPR's command line, as show here:

```
lpr -P lw5 -s test.ps
```

To pass the -s option through APR to LPR when printing classified files, precede it with the double-hyphen sentinel on the APR command line, as shown here:

```
apr -P b113r2222secret -S s -- -s testsrd.ps
```

More Output Control

APR FEATURES.
On a secure machine offering APR, you can specify additional output features by using any of these additonal APR options:

APR option	Output feature
-L	rotate output pages to landscape (132-characters/row)
-2	print output in two columns
-W	truncate lines too wide for the page (default wraps them onto next line)

On LC SCF machines, you can specify which text-to-Postscript converter APR will invoke privately when you run it. Set the environment variable XENSCRIPT to the pathname of the converter that you prefer, where the current choices are:

```
/usr/local/bin/enscript
/usr/local/bin/nenscript  [default]
/usr/local/bin/genscript
```

PREPROCESSORS.
On any UNIX machine, you can use a variety of standard preprocessing tools to control the output characteristics of a text file that you then pipe (|) to LP or LPR for printing. Among the most commonly needed preprocessing sequences are:

Command line	Output feature
nl -ba *file* | lp ...	number each line
pr -d *file* | lp ...	double space all lines
pr -l *n file* | lp ...	set the lines/page to *n*

Test these locally on a small file when you first use them, since implementation details and command interactions often vary by vendor, version, and platform.

Note that you cannont pipe (|) preprocessed files into APR (only into LP or LPR).

USING -o SETTINGS.
On LC OCF Linux machines *only*, you can activate latent output-control features of the Common UNIX Print System (CUPS) by specifying them with the -o option on the execute lines of LP, LPR, and LPOPTIONS. See the next section for details.

Using -o to Control Output (OCF Linux)

ROLE:
On LC OCF Linux machines *only* (such as YANA and ATLAS), you can control many output features by using the -o option to invoke latent aspects of the Common UNIX Print System (CUPS) that manages printing on those machines. You can control output with -o in three (related) ways:

- LP.
  Add one or more invocations of -o to your usual LP execute line to control output for this print job only, such as:

  ```
  lp -d p222 -o landscape -o cpi=12 myfile
  ```

- LPR.
  Add one or more invocations of -o to your usual LPR execute line to control output for this print job only, such as:

  ```
  lpr -P p050 -o job-sheets=standard -o media=letter,transparency myfile
  ```

- LPOPTIONS.
  Add one or more invocations of -o when you run LPOPTIONS to create a permanent .lpoptions file in your home directory that will apply the specified -o features to *every* subsequent LP or LPR print request (that you make on an OCF Linux machine only). For example,

```
lpoptions -p p123 -o sides=two-sided-long-edge
```

SYNTAX:
The examples above reveal that each use of -o takes a *single* argument (although it may in turn have several subarguments), so you must overtly invoke -o several times to specify several different output features on one execute line. Also, the -o argument names are complex and less than intuitive; a glossary of -o arguments is clearly vital for using this approach to control output. Machines where CUPS manages printing offer such a glossary on a local web page:

```
http://localhost:631/sum.html#STANDARD_OPTIONS
```

The simplified, condensed version provides a quick reference for using most -o controls effectively. (The order is roughly alphabetical, except for a few functionally related arguments grouped together for easier comparison.)

ARGUMENTS FOR -o:

brightness=*n*	sets print density (default is 100; greater than 100 is lighter, while less than 100 is darker).
columns=*n*	sets text columns per page (default is 1).
cpi=*n*	sets horizontal characters per inch (default is 10).
lpi=*n*	sets vertical lines per inch (default is 6).
landscape	rotates the text 90 degrees [see also SIDES below].
media=*typelist*	specifies a comma-delimited list of any single *style* (choices are letter, legal, a4, com10 [business envelope], dl [business envelope]), followed by any single printer *tray* (choice of transparency, upper, lower, multipurpose, largecapacity).
mirror	prints mirror image text (for later transfer to other objects).
job-sheets=*bannerstyle*	specifies the presence of and edge labels on a banner page (choices are none [default], standard [banner but no label], classified, secret, topsecret, unclassified).
number-up=*count*	specifies 1, 2, 4, 6, 9, or 16 page images per sheet.
number-up-layout=*string*	specifies the n-up image order in any of eight permutations from btlr (bottom, top, left, right) to tbrl.

page-border=*size*

specifies the framing of each n-up page image (choices are
single [hairline border],
single-thick [1-pt border],
double [two hairline borders],
double-thick [two 1-pt borders]).

output-order=*order*

sets the printing sequence to NORMAL (the default) or REVERSE.

page-left=*n* sets the left margin at *n* points (where each point is 1/72 inch or 0.35 mm).

page-right=*n* sets the right margin at *n* points.

page-top=*n* sets the top margin at *n* points.

page-bottom=*n* sets the bottom margin at *n* points.

page-ranges=*pages*

specifies a comma-delimited list of pages or ranges to print (such as 2,4-6,8,10-12).

page-set=*handed*

specifies printing all (default), ODD, or EVEN pages.

prettyprint displays code listings with automatic feature highlighting.

sides=*duplexflavor*

activates duplex (two-sided) printing in either of two ways:
two-sided-short-edge [duplex landscape],
two-sided-long-edge [duplex portrait].

Night and Weekend Output (B-453, TSF)

To minimize the danger of classified output that lingers unattended, all SCF (secure-network) printer queues that feed printers physically located within B-453, where most of LC's classified production computers reside, are *disabled*:

- Every night starting at 17:00 until the following morning at 7:00, and
- Every weekend starting at 17:00 Friday until 7:00 the following Monday.

WARNING: Print jobs sent to the disabled queues will be purged and *not* held for subsequent output on the next work day.

If you need emergency physical classified output in B-453 when the public SCF print queues are disabled, contact the on-duty computer operator by calling 925-422-0484. Then use the specific non-public SCF print queue that the operator indicates. You must then promptly and personally claim your output at the TSF Operations Center (B-453, Room 2204).

Section 3: Converting File Formats

Print File Conversions

Some graphics file formats may need to be converted into Postscript before they can be printed, including:

- CGM (Computer Graphics Metafile, perhaps the output from GKSGC, GRAFCORE, or NCARLIB graphics libraries)
- DLI (Display List Interactive)
- DVI (standard output from TeX)

After conversion, you can send the file to the current network printers (with LP, LPR, or APR, depending on where the files reside, as shown in the output table above.

Software tools exist to convert each file format to PostScript, including text to PostScript using ENSCRIPT. ImageMagick (URL: https://computing.llnl.gov/vis/imagemagick.shtml) is a popular open-source tool that converts most files to PostScript. Each PostScript conversion tool (except DVIPS) has a MAN-page that details its many control options on the machines on which it runs. But this comparative chart summarizes the basic usage situation, and you can handle most cases just by following the typical command line shown here. In each case, the range option takes starting and stopping frame numbers (*n*, *m*) as its arguments (and if omitted, all frames are processed).

Routine	**Input**	**Machines**	**Typical command line**
cgplot	CGM	IBM/AIX	cgplot *cgmfile*
dvips	DVI	LUCY	-o *outfl.ps infile*
			Range: -pp *n-m*
enscript	text	LUCY	-p *outfl.ps infile*
		Linux machines	(no range control)

CRAY File Conversions

Using data files from CRAY or CDC machines on LC's IBM (AIX) computers often requires that you run a conversion program that understands the older file formats. TRANS on LC's IBMs is one such conversion utility, while LFT is another. This section explains how to use TRANS (on IBMs) appropriately, and it refers you to LFT instructions elsewhere.

TRANS.
In most cases, TRANS can convert text formats and it can also convert numeric values to the IEEE format used on the AIX machines as 8-byte real and integer values. (Note that TRANS cannot deal with real or integer values that do not occupy full words on the older machines. It cannot deal, for example, with byte arrays or bytes packed into structures.) Simple programs can be written to then further convert data to other formats (such as 4-byte integers or reals) as needed.

In many cases, TRANS can figure out what the format of an older file is on its own, but in some cases you must supply extra hints by using TRANS options. The chart below suggests for different source machines and operating systems which TRANS execute line and special options are most likely to work best for specific types of files. Once a file has been converted to IBM format by using TRANS, you can move it to Sun or Compaq machines with FTP, and FTP will automatically handle any futher conversions needed.

For a complete list of all TRANS options (many now obsolete), see the MAN page for TRANS on any LC IBM machine (again, avoid the TRANS description on Compaq machines, which is entirely different).

For files from a CRAY running LTSS, NLTSS, or CTSS

Text editor output,
Fortran sequential formatted write (and namelist and list directed),
Fortran sequential unformatted write,
Fortran direct access formatted write,
Fortran direct access unformatted write,
Fortran two-argument buffer out to a file connected for unformatted I/O,
.....use this TRANS line:

trans -i *infile* -o *outfile*

Fortran two-argument buffer out to a file connected for formatted I/O,
Fortran three-argument buffer out,
.....use this TRANS line:

trans -i *infile* -o *outfile* abs

For files from a CRAY running UNICOS

Text editor output,
Fortran sequential formatted write (and namelist and list directed),
Fortran direct access formatted write,
Fortran direct access unformatted write of character data,
.....use this TRANS line:

[NO translation needed here]

Fortran sequential unformatted write of numeric data,
Fortran two-argument buffer out to file connected for unformatted I/O,
.....use this TRANS line:

trans -i *infile* -o *outfile*

Fortran direct access unformatted write of numeric data,
Fortran two-argument buffer out to a file connected for formatted I/O,
Fortran three-argument buffer out,
.....use this TRANS line:

trans -i *infile* -o *outfile* abs

Fortran sequential unformatted write of character data,
.....use this TRANS line:

trans -i *infile*,cos -o *outfile*

For files from a CDC 7600 or CDC 6600

Most text editor output,
Fortran sequential formatted write (and namelist and list directed),
Fortran sequential unformatted write,
Fortran direct access formatted write,
Fortran two-argument buffer out,
.....use this TRANS line:

trans -i *infile*,7600 -o *outfile*

TRIX Red output,
.....use this TRANS line:

trans -i *infile*,7600,red -o *outfile*

Fortran direct access unformatted write,
Fortran three-argument buffer out,
.....use this TRANS line:

trans -i *infile*,7600 -o *outfile* abs

LFT.

Unlike IBM's TRANS, LFT ("list file types") was written at LC to help manage CRAY and CDC 7600 legacy files. LFT (on all LC production machines) reports and, if you request, converts to UNIX format all specified CRAY and CDC 7600 text files (but leaves all other formats unchanged). See the "Tools for Obsolete File Types" section of EZFILES (URL: https://computing.llnl.gov/LCdocs/ezfiles) for details on LFT and the related tool LIB76.

MOLE (File Review Tool)

MOLE is not really a file-conversion tool, but it serves a similar role if you need to reliably display text files containing nonstandard content. MOLE is a MORE-like text-display utility available to all users on all LC production machines that is customized to help Authorized Derivative Classifiers (ADCs) inspect nonstandard text files proposed for secure-to-open transfer with FIS.

MOLE and MORE are alike in that both programs:

- Display text (ASCII) files at a controlled rate for review (not editing),
- Offer a small command set requiring no carriage return to execute the commands once entered (except for requesting a specific line),
- Use NOECHO mode so that file output is not interrupted by display of the commands that you type (again, except for requesting a specific line), and
- Accept space-delimited lists of files or standard UNIX file filters to display multiple files (one at a time).

But MOLE also offers several LC-developed file-inspection features not supported by MORE:

- CONTROL.
  Added MOLE commands enable forward and backward movement within each displayed file as well as among multiple files. You can also jump to a specified line number in either direction.
- STATUS INFORMATION.
  MOLE reports features of the file being displayed at the start of each new file (in a header) and at the bottom of each screen (in a prompt).
- NONSTANDARD CHARACTERS.
  In files with fewer than 1% non-ASCII characters, MOLE counts and signals each one (with a bell). In files with greater than 1% non-ASCII characters (various binary files), MOLE detects and reports their nontext status without trying to display them. MOLE also counts, signals (with a bell) and displays all "hidden" text lines (lines with a carriage return but no line feed).

If MOLE's extra features meet your file conversion needs, you can see more detailed instructions and annotated usage examples in the MOLE section (URL: https://computing.llnl.gov/LCdocs/fis/index.jsp?show=s6.4) of the FIS Manual or the EZFILES basic guide.

Section 4: Transferring Files Between Machines

File-Transfer Comparison

LC machines, like most UNIX systems, support between-machine file transfers using FTP (file transfer protocol) in a locally parallelized version. Most LC production machines also offer a second, unique file-transfer tool called NFT (network file transfer) that was designed especially with local needs in mind. RCP (remote copy) has been disabled on LC machines because of its security weaknesses, but an alternative tool called SCP (secure copy) with a syntax like that of RCP is available in some situations.

Both FTP and SCP use the TCP/IP network protocols to manage file transfers, and both let you prevent accidental overwriting of existing files (only files with W permission for the owner are overwritten). Also, neither offers queued transfer of multiple files. But their other features diverge, as the comparative chart below shows. When available, SCP is more secure, but FTP is certainly more inclusive since it can transfer files to and from non-UNIX systems (and non-LLNL machines without prior "secure shell" installation), as well. Also, parallel FTP clients are now the default on all LC production machines and no special command is needed to benefit from parallel transfers of large files. Details are available in the FTP Reference Manual (URL: https://computing.llnl.gov/LCdocs/ftp). SFTP (secure FTP) (page 39) behaves much like standard FTP but it encrypts the files that it transfers.

NFT was locally developed and runs only on LC machines where its clients have been specifically installed. Like FTP clients, NFT clients use standard FTP daemons (servers) on remote machines to handle their file transfers. To improve reliability for session tracking and command persistence, however, NFT also employes a third machine running special support software (called ENDEAVOR) to manage NFT jobs. Unlike the other alternatives, NFT offers the convenience of preauthenticated (passwordless) file transfer, and it can be used to transfer files to storage by default and between two remote hosts without logging on to either one. Full details on NFT's features appear in the NFT Reference Manual. (URL: https://computing.llnl.gov/LCdocs/nft)

LC has also developed a separate, specialized file-transfer tool called HTAR, which only moves the members of TAR-format archive (library) files into or out of remote archives located in storage or on other LC production machines. HTAR is fast and efficient, even when there are thousands of member files. A brief section below places HTAR (page 37) in the context of the other tools discussed in EZOUTPUT.

When you run FTP, the network automatically uses these jumbo frame links to transfer your data faster whenever they are available. NFT automatically routes storage-related file transfers from compute nodes to login nodes to take advantage of any available jumbo frame links.

The upcoming subsections give basic usage instructions and examples for SCP (page 25), FTP (page 26), SFTP (page 39), and NFT (page 32). The following table compares the features of these file-transfer alternatives.

Feature	FTP	SCP	NFT	SFTP
Available on:				
.....open machines?	yes	yes(*)	yes(+)	yes(#)
.....secure machines?	yes	yes(*)	yes(+)	no
Logon required?	yes	no	no	yes
Password required?	no	yes	no	yes(#)
Secure shell setup required?	no	yes	no	no
Interactive dialog required?	yes	yes	no	yes
Accepts IP host addresses?	yes	no	no	yes
Confirms successful transfer?	yes	no	yes	yes
Preserves UNIX file permissions?				
.....by default?	no	no	no	no
.....with an option?	no	yes (-p)	no	no
Transfers files between				
.....machines?	yes	yes(*)	yes(+)	yes(#)
.....users?	no	no	no	no
Incoming file overwrites old?				
.....with W permission	yes	yes	no	yes
.....without W permission	no	no	no	no
Supports				
.....queued transfer?	no	no	yes	no
....."persistent" transfer?	no	no	yes	no
.....parallel transfer?	yes	no	yes	no
.....transfers to storage?	yes	no	yes	no
.....transfers to FIS?	yes	no	no	yes
.....use of jumbo frames?	yes	no	yes	no
.....recursive file changes?	no	no	yes	no
Default remote machine?	no	no	yes	no

AVAILABILITY NOTES:

(*)Although SCP executes on all LC machines, you may need to overtly request port 922 (-P 922) on secure machines or else your copy will fail with a misleading "permission denied" message.

(+)Only storage.llnl.gov and those production machines that support NFT clients also accept incoming file transfers using NFT. Other LC hosts (such as FIS (page 42) and some special-purpose SUNs) do not accept NFT transfers.

(#)SFTP *clients* are widely available on OCF machines, but FIS (page 42) currently has the only SFTP server. See the SFTP (page 39) section for optional passwordless use.

Using SCP

SCP Syntax and Limitations

Secure copy (SCP) mimics remote copy (RCP) in its syntax, which has the general form:

```
scp [-P 922] [opts] [[user@]host1:]file1 [[user@]host2:]file2
```

where

-P 922	Specifies port 922. You can omit this option if your local machine (on which you run SCP) defaults to port 922. In general, open LC machines do default to port 922 and secure LC machines do not. Omitting this if needed will yield a (somewhat misleading) "permission denied" message when you run SCP.
opts	SCP control options. The most useful option is -p, which preserves the copied file's original UNIX permissions that get lost by default.
user@	Defaults to your log-on name on your local machine (where you run SCP). You can always omit this unless your log-on name differs between the local and remote machines (e.g., jasons on one and jsmith on the other).
host:	Defaults to your local machine (where you run SCP). You can always omit this on one of the file names on SCP's execute line (whichever is local). The SCP examples below show the simplification that results when you omit the commonly unneeded parts of the SCP execute line.

You should keep in mind several known limitations of SCP as implemented at LC because these contraints affect your ability to transfer files by using SCP locally:

- SCP only works between machines that have a "secure shell" SSH previously installed and enabled for you. This is because SCP requires various authorization and control files in your ~/.ssh directory.

  From outside the llnl.gov domain, only SCP runs begun during an IPA-authenticated session are allowed through the firewall to LC production machines. See the "IPA Authorization" section of LC's Firewall and SSH Guide (URL: https://computing.llnl.gov/LCdocs/firewall/index.jsp?show=s4.3) for instructions; a 2-hour time limit applies.
- SCP treats host names literally, so it cannot tell that GPS01 and GPS01.LLNL.GOV are synonyms for the same machine. This will cause extra dialog with you (shown in the first example below) whenever you change the name by which you refer to a remote machine.
- SCP does not support "third-party" copies (i.e., if you run SCP on one machine, you cannot copy files between two other remote machines). Attempts to copy between two remote machines fail with a somewhat misleading "host identification has changed" warning.
- SCP does not support file transfers to another user. Attempts to copy a file to (or from) a user other than yourself end when SCP prompts you for the other user's password. Use GIVE instead.

- LC's storage system (open and secure) does not accept file transfers using SCP. Store files using FTP or NFT instead.
- LC's File Interchange Service (FIS (page 42)) does not accept file transfers using SCP. Use FTP instead.

SCP Examples

(1) First Outward Copy.

The first time you copy a local file (proj3) to a remote machine that has been enabled for SSH, such as atlas.llnl.gov, an extra dialog is required. Note: This example assumes default use of port 922 and the same user name on both the local and remote machines.) SCP will NOT accept 'y' for 'yes' in this dialog; type the whole word.

```
User: scp proj3 atals.llnl.gov:/var/tmp/jsmith/proj3
Rtne: Host key not found from the list of known hosts.
R/Us: Are you sure you want to continue connecting (yes/no)? yes
Rtne: Host 'atlas.llnl.gov' added to the list of known hosts.
R/Us: jsmith's password: [does not echo when you reply]
      [copy occurs with no further confirmation]
```

(2) Subsequent Outward Copy.

To copy a local file (proj4.ps) to a previously used remote machine (e.g. atlas.llnl.gov), use this brief dialog (same assumptions as above):

```
User: scp proj4.ps atlas.llnl.gov:/var/tmp/jsmith/proj4.ps
R/Us: jsmith's password: [remote password does not echo]
```

(3) Subsequent Inward Copy.

To copy a remote file (e.g. mywork on LUCY) to the local machine (where you run SCP), use this brief dialog (same assumptions as above). The new local file here is called proj5:

```
User: scp lucy.llnl.gov:/var/tmp/jsmith/mywork proj5
R/Us: jsmith@lucy's password: [does not echo]
```

(4) No Defaults.

Same as (3) except make the port explicit (-P922), preserve the original file's UNIX permissions on arrival (-p), and assume the local and remote user names differ (remote is jsmith, local is something else):

```
User: scp -P922 -p jsmith@lucy.llnl.gov:/var/tmp/jsmith/mywork proj5
R/Us: jsmith@lucy.llnl.gov's password: [does not echo]
```

Using FTP

Basic FTP Commands

FTP is a widely used file-transfer utility because it supports transfers between any machines that recognize the TCP/IP protocols, even if they have different architectures or operating systems. You must, however, log in to the remote machine and transfer the files interactively, using your (remote) password. Parallel FTP clients and parallel file transfers are now the default on all LC production machines (see the FTP Reference Manual (URL: https://computing.llnl.gov/LCdocs/ftp) to learn how to invoke nonparallel clients). SFTP ("secure FTP") is a special-purpose, locally modified version of FTP that sends files encrypted but only to dedicated servers (see the SFTP section (page 39)).

FIREWALL ALERT:
LLNL uses a hardware/software security firewall to block direct FTP connections from machines outside the llnl.gov domain to LC machines within llnl.gov. Such FTP blocking means that you must either install special software on your external computer to borrow an llnl.gov IP address (called VPN or Virtual Private Network) or log on to an llnl.gov machine first and then run FTP there to draw remote files toward it. See the Firewall and SSH Guide (URL: https://computing.llnl.gov/LCdocs/firewall) for details and alternatives.

CLIENT DIFFERENCES FOR HPSS ACCESS.
One common use for FTP on LC machines is to put files into or get files from archival storage (HPSS, storage.llnl.gov). But not all FTP clients interact equally well with HPSS. If you work on LC Linux/CHAOS systems (CHAOS 3.0 or later), you have access to /usr/kerberos/bin/ftp, but you should instead run /usr/bin/ftp for reaching storage. Under some conditions, the former (but not the latter) client refuses to log you into HPSS or needlessly asks you to "please log in with USER and PASS."

COMMANDS:
Most FTP implementations support many commands, but not always the same ones. The standard FTP commands, with their syntax and error codes, are discussed in the FTP Reference Manual (URL: https://computing.llnl.gov/LCdocs/ftp). The following FTP commands are the most common, and the crucial ones for basic file transfer:

cd *pathname* — changes directories (on the remote machine) to the one specified by *pathname*. By default, FTP GETs files from and PUTs files to the home directory of the remote machine, so you must change directories with CD if you need to transfer them from or to somewhere else.

pwd — reports the current working directory's pathname on the remote machine (to confirm uses of CD).

dir — lists the names and attributes of files in the current working directory on the remote machine.

get *remotefile* [*localfile*]

retrieves *remotefile* and places it in the current directory of the local machine (where you are running FTP). The incoming file is called *remotefile* by default, or called *localfile* if you specify a name.

mget *filelist* — generalizes the GET command to transfer all the files in *filelist*, a blank-delimited list of remote files to retrieve to the current directory (where you are running FTP). MGET accepts wildcards and prompts for your Y[ES] or N[O] response to each file name before the corresponding transfer.

parallel [LLNL only] reports the current parallel-transfer stripe width and block size (formerly used to enable parallel file transfers, which are now automatic whenever they are available).

put *localfile* [*remotefile*]

copies *localfile* into the home directory of the remote machine you have logged in to with FTP. The outwardly transferred file is called *localfile* by default, or called *remotefile* if you specify a name.

mput *filelist* generalizes the PUT command to transfer all the files in *filelist*, a blank-delimited list of local files to copy to the home directory of the remote machine that you logged in to with FTP. MPUT accepts wildcards and prompts for your Y[ES] or N[O] response to each file name before the corresponding transfer.

help [*command*]

lists the commands supported by the implementation of FTP that you are running or briefly describes one specified command.

quit closes your remote session and terminates FTP.

FTP Examples

The following sample session (with annotated steps added along the right side) shows a typical dialog by which a user (JANE) transfers files interactively using FTP. In this case, the local machine (on which Jane executes the FTP client) is ATLAS, and the remote machine that files are copied to and from is LUCY.

```
(1) The user runs FTP (on ATLAS) with the remote machine's domain
    name as an argument.
(2) FTP prompts for a userid and a password to log in to LUCY
    (some LC machines "preauthenticate" and skip this password step).
(3) At the ftp> prompt, the user changes remote directories
    to /var/tmp/jane (which is not shared among LC machines).
(4) At the next ftp> prompt, the user GETs file NFT.PS
    (copies it from LUCY to ATLAS).
(5) At the next ftp> prompt, the user PUTs file TESTFILE
    (copies it from ATLAS to LUCY).
(6) The user then transfers a 1.1 GB file called LARGE from
    ATLAS to LUCY.  FTP automatically invokes 4 parallel stripes
    (each separately reported as FTP "completed" commands in
    the output from this step).
(7) When the file transfers are done and confirmed, the user
    QUITs FTP.


ftp lucy.llnl.gov                                          ---(1)

Connected to lucy.llnl.gov.
220 [NOTICE TO USERS -- very long legal statement]
222 lucy.llnl.gov FTP server (Version LLNL-22...) ready.
202 Command not implemented.
Name (lucy.llnl.gov:jane): jane                            ---(2)
331 Password required for jane.
Password: [does not echo]
230 User jane logged in.
Remote system type is UNIX.
Using binary mode to transfer files.

ftp> cd /var/tmp/jane                                    ---(3)
250 CWD command successful.

ftp> get nft.ps                                          ---(4)
200 PORT command successful.
150 Binary data connection for nft.ps (134.9.1.5,1863) (1602470 bytes).
226 Binary Transfer complete.
1602470 bytes received in 0.579 seconds (2.64 Mbytes/s)

ftp> put testfile                                        ---(5)
200 PORT command successful.
150 Binary data connection for testfile (134.9.1.5,1877).
226 Transfer complete.
5264 bytes sent in 0 seconds (5.14 Kbytes/s)
```

```
ftp> put large                                                  ---(6)
200 Command complete (11496780, large, 0, 14, 4194304)
200 Command complete.  Address 1 is 134.9.50.35.2356
200 Command complete.  Address 2 is 134.9.50.35.2357
200 Command complete.  Address 3 is 134.9.50.35.2358
200 Command complete.  Address 4 is 134.9.50.35.2359
150 Transfer starting.
226 Transfer complete. (moved = 11496780).
11496780 bytes sent in 0.71 seconds (16.3 Mbytes/s)
200 Command complete.

ftp> quit                                                       ---(7)
221 Goodbye.
```

Passwordless FTP to STORAGE (Only)

Because of the need for fast, reliable file tranfers to and from LC's archival storage system, storage.llnl.gov uses special FTP servers and other LC machines use special FTP clients that can preauthorize your FTP login to storage (only) so that no password is requested. All LC open and secure production clusters now offer passwordless FTP access to storage. The usage is the same as standard FTP (although messages are more verbose), except for omiting the password request. Note also that on LC production machines, NFT, which favors the storage system in several ways, also offers passwordless file transfer to and from storage (see the NFT section (page 32) for details).

Parallel FTP Summarized

PARALLEL CLIENTS.
On all LC production machines, executing FTP automatically executes the parallel FTP (PFTP) client. You can also execute PFTP directly, with the same result. To run the serial FTP client instead, type FPT.BSD (or see other special instructions in the FTP Reference Manual (URL: https://computing.llnl.gov/LCdocs/ftp)).

PARALLEL TRANSFERS.
Parallel file transfers are available among LC production machines, both to and from storage. For files over 1 MB, parallel FTP transfers are ON by default. LC's locally parallelized FTP clients (see above) all support an extra PARALLEL command, but this now merely reports the current parallel block size and stripe width (or number of simultaneous parallel transfers). Actual parallel transfers are automatic whenever they are available.

PFTP COMMANDS.
PFTP offers nine extra commands (beyond the usual set offered by FTP) to specifically manage parallel file transfers (for example, PGET and MPGET perform parallel GETs). On LC production machines these special PFTP commands are quite unnecessary because parallel file transfers are automatic now. At other (ASC Tri-Lab) sites, however, you may need to remember the special PFTP commands to perform parallel file transfers (especially to storage). See LC's HPSS User Guide (URL: https://computing.llnl.gov/LCdocs/hpss) for details on the nine extra PFTP commands.

JUMBO FRAMES.
Jumbo-frame links (page 23) are often available among the same LC machines that support parallel file transfers, so you may be able to benefit from both in cases where that is possible. NFT (page 32) actually "routes" storage-related file transfers to cluster login nodes from compute nodes to take advantage of login-node jumbo-frame links.

HTAR.
LC's locally developed HTAR (page 37) program is a special-purpose front end for PFTP that exclusively transfers member files into or out of stored archives using parallel FTP service behind the scenes.

Using NFT

NFT's Command Syntax

NFT (unlike FTP and SCP) was designed for the LC environment and has two special features affect how you can use it.

First, all NFT file transfers involve not only the donor and the receiver machines you specify (overtly or by default) but also a third invisible machine running locally developed software dedicated to failure detection and recovery, as well as NFT job tracking. Even if problems prevent a file from being sent or received immediately, NFT remembers the request and persists in trying to complete it later, recording its results for you to verify if needed.

Second, NFT assumes that the remote host in all file transfers is the LC storage system (storage.llnl.gov) unless you specify otherwise. The existence of such a default remote host means that:

- There are two types of NFT commands (those for general file transfers among any hosts that NFT serves and those that do not accept NFT's usual host-specifying syntax because they default to storage transfers).
- The syntax of NFT commands assumes local-to-storage transfers unless you specify otherwise.

With FTP (keyword: ftp-examples (page 29)), you always open a connection to a specific target machine BEFORE any file transfers can occur. With SCP (keyword: scp-examples (page 26)), you instead specify the desired source and sink machines with a flag syntax during each separate file transfer. NFT combines these two approaches. NFT's syntax for general file-transfer commands resembles SCP's but is adjusted to reflect the fact that NFT's default target machine is storage.llnl.gov, NOT (as with SCP) the (local) machine on which you are running the NFT client. However, NFT also enables you to change the default target machine with its OPEN command, so that you can (optionally) make it mimic connection-oriented FTP if you wish.

NFT has many special features that suit it for use in batch jobs and scripts (unlike FTP). The NFT Reference Manual (URL: https://computing.llnl.gov/LCdocs/nft) explains those and other unique and unusual features, while its basic syntax and commands are summarized here.

GENERAL SYNTAX:

For NFT, you usually specify the location of a file (or the intended venue of a reporting command like DIR) by means of this threefold colon-flag syntax:

Syntax	Example	Refers to
:	:file1	local (NFT-client) machine
host:	atlas:file1	specified remote machine
	file1	default remote machine, storage unless you change it (NOT local machine)

Using this NFT syntax you can transfer (CP) files to a remote host, from a remote host, or even between two remote hosts ("third-party" transfers), as the NFT Commands by Task chart below shows. This may include transfers to and from storage if you wish.

STORAGE-DEFAULTED COMMANDS:

NFT reserves its PUT and GET commands (and a few others) exclusively for transfers to and from its default remote host (which is storage, unless you change it). Consequently, you never need to (nor can you) use the colon-flag syntax above for literal PUTs and GETs. By using NFT's OPEN command you can redirect PUT and GET to a nonstorage remote host. Subtle but important differences between FTP and NFT's OPEN persist, however, which you should study (URL: https://computing.llnl.gov/LCdocs/nft/index.jsp?show=s9.28b) before you rely on it.

CLASS OF SERVICE (COS):

NFT offers a separate SETCOS command with which you can specify your "class of service" (and hence number of stored copies) for files transferred to storage. SETCOS (URL: https://computing.llnl.gov/LCdocs/nft/index.jsp?show=s9.34b) works easily in job-control files and its results can be checked with NFT's STATUS and DIR -h commands.

NFT Commands by Task

This chart shows the interactive NFT commands that perform the most common file-transfer tasks. For a detailed list of every NFT command, see the Command Dictionary (URL: https://computing.llnl.gov/LCdocs/nft/index.jsp?show=s9) in the NFT Reference Manual.

File-Transfer Task	NFT Command
GENERAL	
Change remote directory to *newdir* on aaa	cd aaa:*newdir*
Change local directory to *newdir*	cd :*newdir*
Change storage(*) directory to *newdir*	cd *newdir*
List remote directory contents on aaa	dir aaa:
List local directory contents	dir :
List storage(*) directory contents	dir
Put (copy) local file *t1* to remote host aaa as *t2*	cp :*t1* aaa:*t2*
Get (copy) remote file *t3* from aaa as local file *t4*	cp aaa:*t3* :*t4*
Transfer file *t5* on aaa to file *t6* on bbb (both remote)	cp aaa:*t5* bbb:*t6*
STORAGE DEFAULTED(*)	
Store local file *t1*	put *t1*
Store local file *t1* as *t2*	put *t1 t2*
Retrieve from storage file *t3*	get *t3*
Retrieve *t3* as local file *t4*	get *t3 t4*
Specify storage class of service (COS) and hence number of stored copies	setcos *nnn*
CONTROL OPTIONS	
Prevent all overwriting (default)	noclobber
Allow overwriting (for updates)	clobber
Start a log of NFT actions	log *logfile*
Close the log file	clog
Change default remote host(*)	open *host*
Terminate NFT	quit

(*)You can change NFT's default remote host from storage to something else by using the OPEN command, but you should read about (URL: https://computing.llnl.gov/LCdocs/nft/index.jsp?show=s9.28b) its behavior before you rely on it.

NFT Examples

This annotated example shows typical file transfers using NFT.

GOAL: To transfer several files among LC machines without logging on to all of the machines, using NFT.

STRATEGY: (1) Start NFT. Notice that UNlike FTP, you do not log on to any particular remote host to "open a connection."
(2) For convenience, change the working directory on ATLAS to /VAR/TMP/STUFF (you could use pathnames later and skip this step).
(3) Transfer (outward copy) local file t1 to /var/tmp/stuff/t2 on ATLAS.
(4) Transfer (inward copy) file /var/tmp/stuff/t3 from ATLAS to local file t4.
(5) Without logging on to either ATLAS or YANA, transfer (copy) /var/tmp/stuff/t3 from ATLAS to /var/tmp/t6 on YANA.
(6) Use storage-defaulted command PUT to transfer file t1 from the client machine (where NFT runs) to storage.llnl.gov as file t2. Note that NO hosts are specified in this command because everything is defaulted.
(7) Try to retrieve file t6 from storage to local file t4 using the storage-defaulted GET command. Becaue NFT's default environment is NOCLOBBER, this attempt fails (t4 already exists as a result of step (4) above). You could use the CLOBBER option next, to allow this overwrite, or...
(8) Use GET to retrieve t6 from storage with no name change (and hence no overwriting of file t4).

```
nft                                          ---(1)

nft> cd atlas:/var/tmp/stuff                 ---(2)
  remote host gps17: wd is /var/tmp/stuff

nft> cp :t1  atlas:t2                        ---(3)
  1.0. 4483 bytes sent in 0.00 seconds
  (4.48 Mbytes/s) from /var/tmp/jfk/t1
  to /var/tmp/stuff/t2
  1.0. 1 entry copied /var/tmp/jfk/t1

nft> cp atlas:t3 :t4                         ---(4)
  2.0. 13081 bytes sent in 0.00 seconds
  (13.08 Mbytes/s) from /var/tmp/stuff/t3
  to /var/tmp/jfk/t4
  2.0. 1 entry copied /var/tmp/stuff/t3

nft> cp atlas:t3 yana:/var/tmp/t6            ---(5)
  3.0. 13081 bytes sent in 0.00 seconds
  (13.08 Mbytes/s) from /var/tmp/stuff/t3
  to /var/tmp/t6
  3.0. 1 entry copied /var/tmp/stuff/t3

nft> put t1 t2                               ---(6)
  4.0. 4483 bytes sent in 0.00 seconds
```

```
  (4.48 Mbytes/s) from /var/tmp/jfk/t1
  to ~/t2
  4.0. 1 entry copied /var/tmp/jfk/t1

nft> get t6 t4                              ---(7)
  5.0. error. lucy.llnl.gov:
  Cannot clobber existing sink.
  /var/tmp/jfk/t4

nft> get  t6                                ---(8)
  6.0. 4585 bytes received in 0.11 seconds
  (41.67 Kbytes/s) from ~/t6
  to /var/tmp/jfk/t6
  6.0. 1 entry copied ~/t6
nft> quit
```

Using HTAR

ROLE.

Unlike FTP, NFT, or SCP, HTAR is *not* a general-purpose file transfer program. It was developed by LC and installed on LC production machines specifically to transfer member files into or out of remote TAR-format archive files. The default location for those archives is HPSS. LC's open or secure permanent storage system, but you can also specify *any* LC production machine as the archive location. HTAR is very efficient, but it cannot transfer files generally, nor to/from) non-LC hosts because it requires a preauthenticated, "passwordless" FTP server.

FEATURES.

HTAR's specially tuned features include:

- A TAR-like command-line syntax and support for TAR-compatible archive files by relying on the POSIX 1003.1 TAR file format.
- Bundling files in memory using multiple concurrent threads and transferring them into an archive file built *directly* in storage (or on another specified remote LC machine), to avoid needing extra local online disk space to build the archive before transferring it.
- Taking advantage of available parallel interfaces to storage (PFTP) to provide fast file transfers (measured at as high as 150 Mbyte/s, or over 30 times the typical rate for transferring small files separately).
- An external index file to easily accommodate thousands of small files in any archive and support retrieval of specified files from within a remote archive without first retrieving the whole archive from HPSS (or from another remote machine).
- Imposes no limit on the total size of the archives that it builds (some have reached 200 GB successfully) and accepts input files (archive members) as large as 8 GB.
- Allows easily building (with -n) and storing incremental archives (containing only recently changed files).

EXECUTE LINES.

When the storage system (HPSS) is up and available to users, you can execute HTAR with a command line that has the general form

```
htar action archive [options] [filelist]
```

and the specific form

```
htar -c|t|x|X -f archive [-BdEFhHILmMopSTvVwY] [flist]
```

where exactly one *action* and the *archive* are always required, while the control options and the *filelist* (except when using -c) can be omitted (the options can share a hyphen flag with the action for convenience). Users familiar with TAR can guess how to run HTAR from this model (although there are some tricky syntax differences). Others should consult the HTAR Reference Manual (URL: https://computing.llnl.gov/LCdocs/htar) for usage suggestions, annotated examples, technical tips, full option details, and known problems.

One very useful special case involves running HTAR to create an archive of files on a remote LC production machine *other than storage* or to extract files from within such a nonstorage archive. HTAR's -F option overrides its storage default (HPSS) and specifies your preferred *target* (machine name) for locating the remote archive (e.g., Yana17):

htar -c|t|x|X -f *archive* -F *target*

See the HTAR manual for an example of this technique and its pitfalls.

HOPPER FRONT END.

You can use LC's graphical file-transfer tool (controller), called HOPPER, to run HTAR as a controllee if you wish to select your files by mouse click. To do this, you must "copy" all of your target files to HOPPER's clipboard, since "Make HTAR Archive" is only available as an operation (submenu choice) on clipboard entries. See the HOPPER section of the HTAR Reference Manual (URL: https://computing.llnl.gov/LCdocs/htar/index.jsp?show=s.25) for details. To list the contents of an existing HTAR archive, find the archive in a HOPPER directory table and *double* click on its icon. Note that unlike HTAR itself, HOPPER never includes the archive's checksum file in its contents report. To extract from an existing HTAR archive, copy the desired files to the CLIPBOARD.

Using SFTP (Secure FTP)

Unlike FTP, NFT or SCP (but somewhat like HTAR), Secure FTP (SFTP) is *not* a general-purpose, general-destination file transfer program. Standard FTP clients do not encrypt the data that they send to remote hosts. The SFTP clients behave generally like FTP, but it *does* encrypt all of the files that it sends for greater protection against third-party eavesdropping. However, SFTP clients interact successfully only with a special SSH2D server daemon and not with the usual FTPD or WU-FTPD servers found on most LC machines.

So SFTP provides extra file-transfer security, but with these significant limitations:

- SFTP clients reside only on OCF (open) LC machines, not on SCF machines.
- The only currently supported SFTP server that accepts incoming files from SFTP clients is on the OCF File Interchange Service (FIS) node at fis.llnl.gov. In particular, you *cannot* send files to storage (storage.llnl.gov) by using SFTP.
- SFTP supports only about half of the standard FTP control options. In particular, the options ASCII, BIN, DELETE, and DIR are not supported by SFTP.

See the SFTP section (URL: https://computing.llnl.gov/LCdocs/ftp) of LC's FTP Reference Manual for a more complete discussion of SFTP features and limitations, for a comparison of FTP and SFTP user dialogs, for suggested ways to work around the missing SFTP control options, and for information on using "DSA keys" instead of your one-time password (OTP) to authenticate SFTP sessions (to FIS).

Using HOPPER (Controller)

SERVICES:

HOPPER offers an alternative, graphical interface to (a GUI controller for) these file-transfer services at LC:

- FTP, SFTP, and NFT (on its CONNECT menu, as a pull-down on the CONNECT TO REMOTE option),
- HTAR (on its CLIPBOARD menu to create an archive; *double* click on an existing archive's directory entry to report its contents, then copy wanted items to HOPPER's CLIPBOARD),
- "Storage HPSS" (really opens an FTP connection to your top-level directory at storage.llnl.gov by default), and
- File Interchange Service (FIS) for transfers between LC's open and secure networks.

FEATURES:

By invoking HOPPER, you can do many file-management tasks graphically, including:

- Use one common, desktop-like interface to run a variety of otherwise disparate file-handling tools.
- Avoid learning the syntax and specific options for tools that are usually executed by a UNIX-style command line with explicit arguments.
- Pick visually an *alphabetically scattered* set of files to transfer from a larger set, where file filters won't help.
- Pick *graphically* from a set of already HTAR-archived files a subset to extract (sometimes tricky with a command line).

HOPPER's SELECT menu also offers a USE WILDCARD option that gives you the convenience of a file filter within its graphical framework to accelerate those cases where filtering meets your file-selection needs.

STRATEGY:

HOPPER is written in Java. It has been installed on all LC production machines, open and secure, and you can find instructions on the support Web site (below) for downloading the source for use on desktop computers. You use HOPPER by:

- Executing the program,
- Picking graphically (CTRL-CLICK with your mouse) the *files* on which to operate from its displayed list or table of those in your chosen directory, and then
- Specifying (from a menu) which *operation* to perform on those files (such as put into or retrieve from a remote directory). You can also define a set of UNIX commands in advance and then "launch" any of them as (additional, customized) operations on your selected target file(s). See the LAUNCH COMMANDS subsection below for instructions.

PERFORMANCE:

Because transferring many files and displaying much graphical data (both, perhaps, remotely) can in some circumstances slow performance, you will find that using HOPPER is more satisfying the closer to the top of this spectrum of conditions you can arrange to execute it:

- Ideal: run HOPPER on your local desktop machine (where you want it to display), even when moving files on remote production hosts.
- Adequate: connect to a remote production host by using SSH on your local desktop machine, and run HOPPER remotely to display back to your local host over X11. On Windows computers, use XWin-32's built-in SSH.
- Least favorable: connect to a remote production host by using F-Secure SSH on your local desktop, and run HOPPER remotely to display back to your local host over X11.

LAUNCH COMMANDS:

To expand the set of operations that HOPPER lets you perform on a file, HOPPER includes the ability to define a customized set of UNIX commands and then (later) execute any of those commands on a target file (so long as the target file resides on a UNIX system, even if you run HOPPER elsewhere).

To define a customized launch command, work down this chain of menus starting at FILE:

- File > Preferences > Operations > Launch Command
- Fill in the displayed dialog box with a mnemonic label ("command name," shows in a later menu) and "command line" (exact UNIX syntax to execute). Use the string %s to parameterize the command. For example, supplying name PRINT and line

  ```
  lp -d p280 %s
  ```

  lets you (later) click on a PRINT menu entry to execute the above specific UNIX printer command on your current HOPPER target file. Click on the ADD button to save each newly defined command.

"Launching" a previously defined command:

- Right click the target file from HOPPER's directory table or choose Ops on Selected Entries from HOPPER's OPS menu.
- In the drop-down menu that appears, go to Launch Command and choose the command you want to launch.

SUPPORT:

More general background information on HOPPER is available at the (OCF-only) project Web site:

`http://computing.llnl.gov/hopper`

See "Getting Started" for instructions on how to download HOPPER to your local desktop machine.

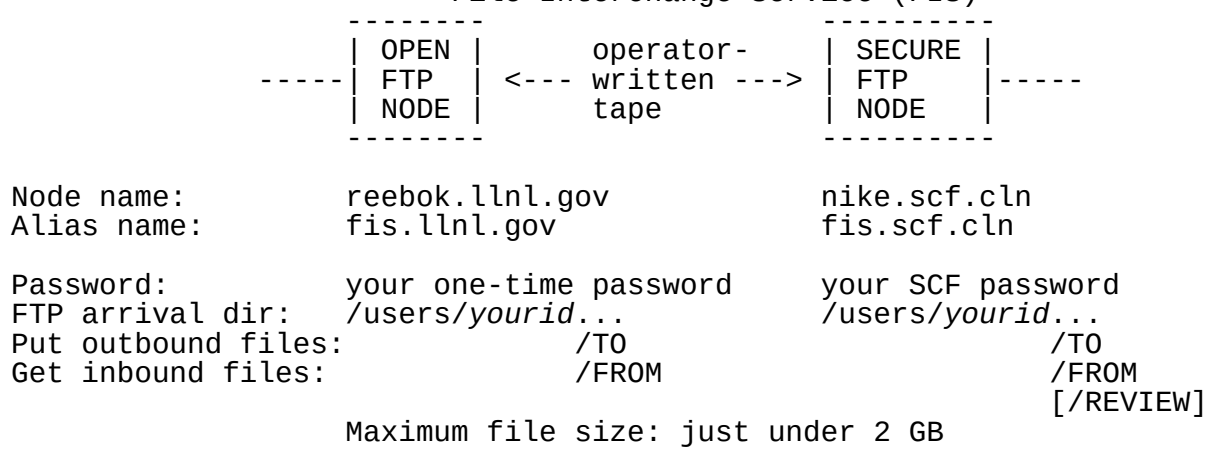
Section 5: Open/Secure File Interchange Service (FIS)

FIS Overview

LC's open and secure networks are physically isolated from each other, but you can transfer files between them by using FTP and two specially designated transfer nodes. You can FTP the file(s) to the outbound directory of one network's transfer node, an LC operator moves the file(s) on tape between networks, and you retrieve them (with FTP) from the inbound directory of the other network's transfer node.

NFT and SCP will not transfer files to or from FIS. SFTP ("secure FTP") clients on open-network LC machines can transfer files to a special SFTP server on the open-network FIS node. The interaction is much like using standard FTP except that the transferred data is encrypted. But many fewer options are available to control the transfer (see the SFTP (page 39) section for details). SFTP is *not* available on the secure network. HOPPER (page 40) offers a graphical user interface for FTP transfers to or from FIS on LC production machines.

This diagram summarizes the between-network file-interchange process:



Note that while LC now has 36 GB of total disk space devoted to FIS file transfers (on both the open and secure networks) to support many simultaneous FIS users without congestion, the largest *individual* file that you can transfer through FIS is just under 2 GB. See also the "Open to Secure" (page 43) subsection below for special advice about using FIS on weekends.

Authorization

To use the file-interchange service (FIS) you must already have an account and valid password for at least one open and one secure machine. You must also complete the FIS form, which is available from the LC Hotline or online:

```
https://computing.llnl.gov/forms/fis.pdf
```

This form is used to create your FIS account and needs Computer Coordinator approval for open-to-secure transfers. It requires division or department head approval for secure-to-open transfers, which an ADC always monitors. FIS accounts do not expire once approved.

Any user can receive authorization to move files from the open to the secure network. To receive authorization, you must specify the kinds of files to be moved and obtain the approval of your division leader or department head (using the appropriate form). See the instructions on the form and in the FIS Reference Manual (URL: https://computing.llnl.gov/LCdocs/fis) for full details.

Once you are authorized, your authenticator-generated one-time password (OTP, used for any production open computing cluster) will allow access to the open FIS node as well.

On the secure side, the FIS node uses your secure DCE password (or your secure OTP if you have elected SCF OTP use).

File-Transfer Process

From Open to Secure Network

Once you are authorized and have your password, follow these steps to transfer files from the open to the secure networks:

- Log on to the open machine where your files to transfer reside.
- Run FTP (or, if you prefer, SFTP (page 39)). Type

  ```
  ftp fis.llnl.gov
  ```

  and use your authenticator-generated one-time password (OTP) to start an FTP session to the open FIS node.
- PUT your file(s) into the directory /users/*yourid*/TO on fis.llnl.gov. Note that directory's name is indeed uppercase. (Use FTP's binary mode for TAR files.) FIS converts all nonalphanumeric characters (except embedded dots (.)) in the *names* (not bodies) of transferred files into underscore characters during transfer (see "How FIS Handles File Names" in the FIS Manual (URL: https://computing.llnl.gov/LCdocs/fis) for full details and a work-around).
- Wait about 1 hour while the operators write your files on tape and then read that tape on the secure network. The actual delay depends on the age and size of the files and the amount of file-transfer traffic.
- Log on to the secure machine where you want to receive your files.

- Run FTP. Type

  ```
  ftp fis
  ```

  and use your secure DCE password (or SCF OTP if you have enabled one) to start an FTP session to the secure FIS node.
- GET your file(s) from the directory /users/*yourid*/FROM on fis.llnl.gov. Note that directory's name is indeed uppercase.
- After the arrival of your files on another machine has been confirmed, delete them from the /FROM directory on the secure FIS node to save space.

From Secure to Open Network

Once you are authorized and have your DCE password (or SCF OTP), follow these steps to transfer files from the secure to the open network:

- Log on to the secure machine where your files to transfer reside.
- Run FTP (SFTP is not available on SCF). Type

  ```
  ftp fis
  ```

  and use your secure DCE password (or SCF OTP if you have enabled one) to start an FTP session to the secure FIS node.
- PUT your file(s) into the directory /users/*yourid*/TO on fis.scf.cln. Note that directory's name is indeed uppercase. FIS converts all nonalphanumeric characters (except embedded dots (.)) in the *names* (not bodies) of transferred files into underscore characters during transfer (see "How FIS Handles File Names" in the FIS Manual (URL: https://computing.llnl.gov/LCdocs/fis) for full details and a work-around).
- Since former ADMIN and DUSA usage of FIS has been suspended in favor of the "NEED category," you must always personally contact your organization's local Authorized Derivative Classifier (ADC) to review your files staged for transfer. (Consult the FIS Manual (URL: https://computing.llnl.gov/LCdocs/fis) for technical details on how your ADC carries out this review, often by using ADCTOOL and MOLE.)
- After the review is complete and your ADC approves the transfer, your files will move to the open network by tape (same as above).
- Log on to the open machine where you want to receive your files.
- Run FTP (or, if you prefer, SFTP (page 39)). Type

  ```
  ftp fis.llnl.gov
  ```

  and use your authenticator-generated one-time password (OTP) to start an FTP session to the open FIS node.
- GET your file(s) from the directory /users/*yourid*/FROM on fis.llnl.gov. Note that directory's name is indeed uppercase.
- After the arrival of your files on another machine has been confirmed, delete them from the /FROM directory on the open FIS node.

Disclaimer

Keyword Index

To see an alphabetical list of keywords for this document, consult the next section (page 49).

Alphabetical List of Keywords

Keyword	Description
a	The alphabetical index of keywords.
apr-features	LP, LPR, APR output features compared.
availability	Where these programs run.
big-files	Printing very large files.
classification-labels	APR security-level choices.
date	The latest changes to EZOUTPUT.
default-printers	Specifying defaults with PRINTER, LPDEST.
discovering-printers	How to find a printer.
entire	This entire document.
environment-variables	Specifying defaults with PRINTER, LPDEST.
file-conversion	Converting file formats.
file-interchange-service	Open/secure file-transfer service.
file-transfer	Transferring files between machines.
file-transfer-comparison	SCP, FTP, NFT features compared.
fis	Open/secure file-transfer service.
fis-authorization	Getting permission to use FIS.
fis-overview	Usage diagram, brief strategy.
fis-steps	Open-secure file-transfer steps.
ftp	Using FTP to transfer files.
ftp-commands	Basic FTP options explained.
ftp-examples	Sample file transfers with FTP.
hopper	Using HOPPER to move files graphically.
htar	Using HTAR to transfer archives.
index	The structural index of keywords.
introduction	Role and goals of EZOUTPUT.
listpq	Location-based printer discovery.
local-printers	How to identify local printers.
lp-features	LP, LPR, APR output features compared.
lpr-features	LP, LPR, APR output features compared.
lpstat	Get current status of a physical printer.
mole	Inspecting nonstandard text with MOLE.
nft	Using NFT to transfer files.
nft-commands	Basic NFT options by task.
nft-examples	Sample file transfers with NFT.
nft-syntax	Specifying sec. levs, hosts with NFT.
night-weekend-output	Limits on SCF printing at night.
open-output-example	Deleting a print job.
open-secure-transfer	Open/secure file-transfer service.
open-to-secure	File transfers to secure network.
option-o-control	Controlling output with option -o.
output-comparison	LP, LPR, APR output features compared.
output-format	Format control with APR, PR, NL.
parallel-ftp	Parallel file transfers summarized.
print-conversion	Converting DLI, DVI, CGM to PS for printing.
print-examples	Sample file-printing sessions.
print-queue-names	How LC names print queues.
printcap-files	Printer names in PRINTCAP files.
printing-files	How to print text and PS files.
rcp	Using RCP to transfer files (disabled).
revisions	The complete revision history.
save-output	Printing to a file.
scope	Topics covered in EZOUTPUT.
scp	Using SCP to transfer files.
scp-examples	Sample file transfers with SCP.

Date and Revisions

Revision Date	Keyword Affected	Description of Change
29May09	lpstat	Added command description.
02May07	print-conversion	Compaq cases deleted.
	file-transfer-comparison	NFT now uses jumbo-frame links.
	scp-examples	ATLAS replaces GPS in examples.
	ftp-commands	PARALLEL now for reporting only.
	ftp-examples	Parallel-transfer details updated.
	parallel-ftp	Parallel transfers now automatic.
	nft-syntax	NFT now uses jumbo-frame links.
	nft-commands	NFT now sets class of service.
	nft-examples	All GPS examples updated.
13Dec06	hopper	More usage details and tips.
	fis	HOPPER now supports FIS transfers.
15Nov06	night-output	New section on SCF output limits.
	weekend-output	New section on SCF output limits.
	index	New keywords for new section.
08Aug06	fis-authorization	New OCF URL, no SCF URL.
17Jul06	ftp-commands	Warning added on 2 FTP clients.
	output-comparison	Purple replaces White for APR.
	environment-variables	Cross ref to Env. Var. guide added.
07Dec05	htar	HOPPER omits HTAR checksum files.
	hopper	Launch feature added, explained.
25Jul05	htar	New features noted.
	hopper	Graphical controller explained.
	index	New keyword for HOPPER section.
27Jun05	scp	IPA 2-hour limit, cross ref added.
04Aug04	option-o-control	More (Linux) output control features.
	print-queue-names	New standard printer names.
	discovering-printers	How to find an LC printer.
	listpq	New OCF tool explained.
	index	New keywords for new sections.
24Jun04	introduction	TSB cross ref for graphics output.
	output-format	APR's setable XENSCRIPT variable.
27Apr04	fis-authorization	Form LC0005 replaces former SCF-6 for FIS.

23Feb04	file-transfer-comparison	NFT now has recursive options.
	nft-syntax	New ACL commands noted.
	nft-commands	Five ACL commands added.
	nft-examples	Revised to show more verbose output.
18Nov03	htar	Size and speed details updated.
06Aug03	file-transfer-comparison	HTAR role expanded.
	htar	Between-machine (-F) role added.
01May03	ftp-commands	PARALLEL command role revised.
	parallel-ftp	More automatic parallel transfers.
	ftp-examples	GPS17 replaces GPS01.
	nft-examples	GPS17 replaces GPS01.
04Mar03	fis-overview	Cross ref to weekend warning.
	open-to-secure	Two-person rule, work-around noted.
05Feb03	sftp	New section for new tool.
	index	New keyword for new section.
	file-transfer-comparison	SFTP added to overview, chart.
	ftp	SFTP cross reference added.
	fis	SFTP role explained.
20Nov02	file-transfer-comparison	FTP now monitored by file size.
	ftp-commands	FTP monitored by file size, tasks.
16Oct02	output-comparison	Linux now supports APR etc.
	nft	NFT now under Linux too.
21Aug02	introduction	Cross ref added to I/O Guide.
	fis-overview	Total vs individual file limits.
03Jul02	htar	Y option added.
11Feb02	entire	Names in examples updated throughout.
28Jan02	file-transfer-comparison	NETMON cross reference added.
	ftp-commands	NETMON cross reference added.
	entire	LUCY replaces K2 everywhere.
09Oct01	ftp-examples	Open one-time passwords ok.
	fis	Open one-time passwords ok.
04Sep01	htar	New section on TAR-like tool.
	index	New keyword for new section.
	file-transfer-comparison	HTAR role briefly noted.
	introduction	HTAR role briefly noted.
10Jul01	file-transfer-comparison	Jumbo frame role noted.
	ftp-commands	PARALLEL toggle added.

	ftp-examples	Parallel transfers illustrated.
	parallel-ftp	New summary section added.
	index	New keyword for new section.
21May01	mole	New section on local file tool.
	index	New keyword for new section.
18Apr01	ftp	VPN role noted for offsite.
	scp	IPA authentication needed offsite.
	macintosh-problems	VPN role noted for offsite FTP.
20Mar01	file-transfer-comparison	Default is now parallel FTP client.
	ftp	Parallel FTP client role, verbosity noted.
23Jan01	fis-steps	File-name character changes noted.
24Jul00	security-levels	PARD security option discontinued.
	apr-example	Former PARD example updated.
07Jun00	fis	New secure-to-open instructions.
08Sep99	file-conversion	Subdivided; TRANS use added.
	fis	DCE passwords used now.
	index	New keywords for TRANS.
14Jun99	macintosh-filenames	How to handle names with blanks.
	index	New keyword added.
03Jun99	output-comparison	Meiko (Tribble) entries deleted.
22Feb99	ftp	FTP firewall-blocking enabled.
	macintosh-ftp	New section on Mac problems.
04Feb99	ftp	Firewall alert added.
05Jan99	introduction	SCP replaces RCP.
	file-transfer-comparison	SCP replaces RCP features in table.
	rcp	RCP disabled now.
	scp	Syntax, limits, examples added.
	fis	FTP only, not SCP or NFT.
	index	SCP keywords added.
10Aug98	introduction	Added, clarified scope.
	output-comparison	Borg, Y-MP entries deleted.
	file-conversion	Borg, Coral entries deleted. DEC entries added.
	file-transfer	NFT on open machines too.
	nft	NFT on open machines too.
	ftp	FTP Reference Manual linked.
10Dec97	file-transfer-comparison	Table updated, notes expanded.

	storage	New hosts for passwordless FTP.
	nft-syntax	OPEN role noted throughout.
	nft-commands	OPEN role added to table.
	nft-examples	Storage-defaulted replaces "only."
07Jul97	output-comparison	DEC and IBM table entries updated.
	lpr-example	Example output clarified.
	nft-syntax	Security option disabled.
	nft-examples	Security option removed.
06Jan97	fis	Open/secure file transfers added.
04Sep96	file-transfer-comparison	NFT features, roles added.
	nft	NFT syntax, commands, examples added.
	storage	Passwordless FTP details added.
	index	NFT-related keywords added.
12Aug96	entire	Greatly expanded with new sections on local printers, file conversion and transfer.
30Jul96	entire	First edition of LC EZOUTPUT manual.

ANG (11Jun09)

UCRL-WEB-200048